

From Piecewise Linear Networks to Context-Selected Linear Mappings

Most machine learning researchers are familiar with the observation that ReLU neural networks are **piecewise linear**. Although the network as a whole is nonlinear, every fixed pattern of ReLU activations corresponds to a single linear mapping from input to output.

For readers less familiar with this viewpoint, the reason is straightforward. Each ReLU computes

$$\text{ReLU}(x) = \max(0, x),$$

so it either passes its input unchanged or outputs zero. Once every ReLU has made this binary decision, the remainder of the computation is entirely linear.

A useful observation is that every ReLU decision can be represented by a single binary value:

- 0 if $x \leq 0$
- 1 if $x > 0$

Collecting these binary decisions across a layer produces a **binary context vector** describing the computational state of that layer.

For any fixed binary context vector, the layer performs one particular linear mapping. Changing one or more bits changes the selected mapping.

This immediately suggests a different interpretation of the role of the ReLU.

Rather than viewing the ReLU primarily as a nonlinear function, we may instead view it as a mechanism for constructing a **binary context vector** that selects one linear operator from a large family of possible operators.

The same reasoning naturally extends to the entire network.

If a ReLU network implements one particular global linear mapping for a given input, then that global mapping must be the product of the mappings chosen by the individual layers. There is little alternative: each layer must itself be selecting one of many possible local linear mappings.

The global input therefore does not directly choose a single global operator. Instead, it propagates through the network, successively generating a binary context vector at each layer. Each binary context vector selects that layer's local linear mapping, and the overall computation is simply the concatenation of these selected mappings.

Viewed this way, every ReLU layer consists of two conceptually distinct operations:

1. Construct a binary context vector from the incoming activations.
2. Use that context vector to select a local linear operator.

The computation and the control mechanism are therefore intertwined within the same activation pathway.

This observation suggests a broader architectural principle.

Separate computation from control.

Rather than deriving the context implicitly from the previous layer's activations, future architectures may provide context through explicit control pathways carrying task identity, temporal state, robot operating mode, memory, dialogue state, environmental state, or other contextual information.

The layer then becomes a context-selected linear mapping,

$$y_i = M_i(c_i)x_i,$$

where c_i is the context supplied to that layer.

The complete network becomes

$$y = M_n(c_n)M_{n-1}(c_{n-1}) \cdots M_1(c_1)x.$$

This interpretation shifts the emphasis of deep learning. The primary role of the nonlinearity is not to perform increasingly complicated nonlinear computation. Its primary role is to determine **which linear computation is performed**.

Many existing architectures can be viewed from this perspective. ReLU networks generate context implicitly through activation patterns. Mixture-of-experts architectures generate context through routing decisions. Hypernetworks generate operators from conditioning variables. Conditional computation architectures employ explicit control signals.

These architectures differ principally in **how context is generated**, rather than in the computation performed after the operator has been selected.

Viewing networks through the lens of context selection also opens a wide range of architectural possibilities.

The context supplied to each layer need not originate from the previous layer alone. Different layers may receive different forms of context. One layer may respond to temporal information, another to robot operating mode, another to memory state, another to environmental conditions, and another to task identity. Some context signals may be binary, while others may select among operators using soft weighting or interpolation. Different layers may also operate on very different context timescales.

This separation of data and control greatly enlarges the design space while preserving a simple computational core: a sequence of selected linear mappings.

It also suggests that many apparently different neural network architectures may share a common underlying structure. Their principal differences lie not in the computations themselves, but in the mechanisms used to construct and distribute context throughout the network.

Seen from this viewpoint, **Concatenated Context-Selected Linear Mappings (CCSLMs)** are less a single architecture than a general design framework. They encourage researchers to think explicitly about context as a computational resource in its own right, independent of the linear operators it selects. Once this separation is made, numerous new architectures become possible while retaining the simplicity, interpretability, and compositional nature of linear mappings.